Software Quality Professional

VOLUME ELEVEN • ISSUE THREE • JUNE 2009

ASQ
www.asq.org

MISSION STATEMENT: Software Quality Professional (SQP) is a peer-reviewed quarterly publication applying quality principles to the development and use of software and software-based systems.

It publishes case studies, experience-based reports, and state-of-the-art reviews in order to provide practitioners with an understanding of those software quality practices that have proven effective in a wide range of industries, applications, and organizational settings.

To enhance personal and professional growth, the journal provides a forum for exchanging practical ideas and experiences.

SQP constantly strives to improve the professionalism of practitioners, the satisfaction of customers, and the well-being of the larger society.

SOFTWARE QUALITY PROFESSIONAL (ISSN 1522-0542) is published quarterly (December, March, June, September) by ASQ, 600 N. Plankinton Ave., P.O. Box 3005, Milwaukee, WI 53201-3005. Periodicals postage paid at Milwaukee, WI. Canadian GST #128717618. Back issues are available by calling ASQ at 800-248-1946 (United States or Canada) or 414-272-8575 and are based on availability. ASQ members $15 per copy; nonmembers $23 per copy.

For microform, contact ProQuest Information and Learning, 300 N. Zeeb Road, Ann Arbor, MI 48106, 800-521-0600, ext. 2888, international 734-761-4700, http://www.il.proquest.com.

MANUSCRIPT REQUIREMENTS: Manuscripts should be submitted to the manuscript coordinator as described in the "Guidelines for Authors" on the SQP Web site at http://www.asq.org . Material submitted to SQP must be the original work of the authors, each of whom must have made a substantive contribution to the work. Content must be substantially different from other works previously published or under review by another publication.

POSTMASTER: Please send address changes to SOFTWARE QUALITY PROFESSIONAL, ASQ, P.O. Box 3005, Milwaukee, WI 53201-3005.

No claim for missing issues will be accepted after three months following the month of publication of the issue for domestic addresses and six months for Canadian and international addresses.

Publication of advertisements does not constitute endorsement of any particular product or service by ASQ.

Printed in USA

	ASQ Members	Nonmembers	Institutional
U.S.	$45.00	$75.00	$120.00
International	$70.00	$95.00	$150.00
Canadian	$65.00	$95.00	$150.00

Rates subject to change without prior notification. Institutional rates include first-class delivery. Canadian rates include 7% GST.

Institutional subscriptions are held in the name of a company, corporation, government agency, or library.

http://www.asq.org

Contents

VOLUME ELEVEN • ISSUE THREE • JUNE 2009

DEPARTMENTS

Overview

This issue has articles on software metrics and analysis—and one Talking Points article reminding us what we have learned from past disasters. The Resource Reviews cover three categories of the Certified Software Quality Engineer (CSQE) body of knowledge. We end with a Quality Nugget—"2008 Presidential Election Results Obtained Smoothly Due to Best Practices Approach for Load and Stress Testing" by Stephane Jammet, vice president of Neotys.

The first article is "MIS-PyME Software Measurement Goals Table: Supporting the Selection of Measurement Goals Based on Measurement Organizational Maturity" by María Díaz-Ley, Félix García, and Mario Piattini. Small and medium enterprises (SMEs) have specific problems in successfully implementing measurement programs, since the typical characteristics of these companies do not facilitate this task: limited resources, budget, and training; poor software measurement knowledge; and poor measurement mentality. Consequently, these kinds of settings not only require a methodology that is specifically adapted to the definition of measurement programs in SMEs, but they also need guidelines to facilitate the successful definition and implementation of the measurement program. This article describes one of the support work products, the MIS-PyME measurement goals table, which suggests common measurement goals that are aligned to the corresponding measurement maturity. The article also shows the MIS-PyME measurement goals table's usefulness when applied to a case study.

Next we hear from Yogesh Singh and Anju Saha on "Assessment of Class Testability: An Empirical Study." Testing is one of the most expensive phases of the software development life cycle. Assessment of a software's testability may help in planning testing activities and making the testing process more effective. In this article, some of the factors of testability are identified for object-oriented software. The correlation between source code metrics and test metrics shows that most of the metrics are significantly correlated to test metrics. The results show that testability can be assessed from source code metrics. This study may help software practitioners to focus more on testing classes with low testability. Further, the classes with very low testability can be redesigned to improve their testability.

Then Patricia A. McQuaid helps us to understand "Software Disasters—What Have We Learned?" Over the years, there have been several major software disasters resulting from poor software project management, poor risk assessment, and poor development and testing practices. The results of the disasters include project delays, project cancellations, and human fatalities. This article examines such failures as the Mars Polar Lander, the Patriot missile, and the Therac-25 radiation deaths. The focus of this article is on the factors that led to these problems, an analysis of the problems, and the lessons to be learned that relate to software engineering, safety engineering, government and corporate regulations, and oversight by users of the systems. This article is designed to inform the reader about these tragedies and to point out that in some cases, while the error was minor, the consequences were tragic. In other cases, there were multiple interrelated issues, some masking deeper problems.

Finally, Bindu Goel and Yogesh Singh explain "An Empirical Analysis of Metrics to Predict the Maintainability for Real-Time Object-Oriented Software." Many metrics have been proposed in the literature for measuring different properties of object-oriented software such as size, complexity, cohesion, coupling, and inheritance. In this study, an empirical investigation is done to find the relationship between these metrics and the maintainability of the software. Maintainability can be defined as the ease with which software can be changed. Both the univariate and multivariate models are proposed using a linear regression analysis technique on the data collected from a real-time, large object-oriented system. The results suggest that size, complexity, coupling, and cohesion metrics are significant predictors for measuring maintainability of classes, while inheritance measures are not.

Software Quality Professional

EDITORIAL/PRODUCTION

From the Editor

Welcome to *Software Quality Professional* (*SQP*), volume 11, issue 3. The Overview provides details about the articles in this issue. Remember that *SQP* cannot be a quality journal without the help of many volunteers. If you would like to submit an article, review articles or books, write a letter to the editor, or provide feedback—please e-mail me at sue_carroll@bellsouth.net.

The Software Division is preparing for the International Conference on Software Quality, November 9-11, 2009, in the Chicago area. I hope you can join us. We will be offering a wide variety of tutorials November 9 and a two-day conference November 10-11. See www.asq-icsq.org for program details. In the next issue of *SQP*, I will let you know about the Institute for Software Excellence, which was held in May in conjunction with the World Conference on Quality and Improvement.

I would like to thank Peter Poon for his years of service to *SQP* as an editorial review board member, and I would like to welcome three new editorial review board members: J. David Blaine, Trudy Howles, and Uma Ferrell.

J. David Blaine is an independent consultant with more than 30 years of experience as a software developer, systems engineer, project manager, and improvement specialist. He consults for software process improvement and project management. Blaine earned a master's degree in mathematics and a master's degree in electrical and computer engineering. He was the quality lead for the PMI Guide to the Project Management Body of Knowledge 2004 Update Project. Blaine sits on the Advisory Board of *IEEE Software* magazine and is a member of ACM, IEEE, PMI, and ASQ. He is accredited with the PMPR and ASQ CSQE, CPQA, and CMQ/OE.

Trudy Howles is an associate professor of computer science at the Rochester Institute of Technology in Rochester, NY. She is a CSQE. Her research interests include programming and quality skills taught at the college level, and student persistence and interest in computing disciplines. She has published several related articles with ACM, IEEE, and ASQ. Prior to entering the teaching field, Howles worked in industry for 25 years. She has held positions as a senior software engineer, project manager, and consultant at Bankers Trust, Eastman Kodak, the former Digital Equipment Corporation, and General Motors.

Uma Ferrell (formerly Satyen) is a cofounder of Ferrell and Associates Consulting, Inc., serving suppliers of safety and mission-critical software-intensive systems. Ferrell's accomplishments include work in system and software safety, development assurance, standards and process development, and aircraft certification. Ferrell coauthored IEEE 1012 (Software Verification and Validation) and RTCA DO-178B (Software Considerations in Airborne Systems and Equipment Certification). She has authored material for both the *CRC Handbook on Avionics* and the *Wiley Encyclopedia of Software Engineering*. She has been an *SQP* article reviewer since 2003. Ferrell holds a master's degree in electrical engineering from Johns Hopkins University. She also holds a master's degree in solid-state physics and bachelor's degrees in physics, chemistry, and mathematics from Bangalore University in India.

Sue Carroll

MIS-PyME Software Measurement Goals Table: Supporting the Selection of Measurement Goals Based on Measurement Organizational Maturity

MARÍA DÍAZ-LEY
STL

FÉLIX GARCÍA AND MARIO PIATTINI
University of Castilla

Small and medium enterprises (SMEs) have specific problems in successfully implementing measurement programs, since the typical characteristics of these companies do not facilitate this task: limited resources, budget, and training; poor software measurement knowledge; and poor measurement mentality. Consequently, these kinds of settings not only require a methodology that is specifically adapted to the definition of measurement programs in SMEs, but they also need guidelines to facilitate the successful definition and implementation of the measurement program. The MIS-PyME framework provides a methodology that has been specially adapted for the definition and implementation of measurement programs in SMEs, and that also provides support modules to guide the user in this task. This article describes one of the support work products, the MIS-PyME measurement goals table, which suggests common measurement goals that are aligned to the corresponding measurement maturity. These can be used to achieve common process improvement goals and to define measurement programs more efficiently, effectively, and accurately. The article also shows the usefulness of the MIS-PyME measurement goals table when applied to a case study.

Key words

measurement program definition, MIS-PyME measurement goals table, process improvement goals, SMEs

SQP References

Use of Metrics in High Maturity Organizations
vol. 4, issue 2
Pankaj Jalote

Measurement Maturity and the CMM: How Measurement Practices Evolve as Processes Mature
vol. 4, issue 3
Charles Weber and Beth Layman

INTRODUCTION

Software measurement and analysis is widely recognized as an effective means to understand, monitor, control, predict, and improve an organization's software development and maintenance projects, products, and processes. Effective software measurement, however, requires the documentation of a lot of information, models, and decisions. It is therefore a difficult task for people who do not have extensive experience in software measurement (Briand, Differding, and Rombach 1996). Defining and implementing software measurement programs continues to be an intellectually complex process, which requires a considerable amount of time, effort, and experience (ESSI 1996). This is particularly difficult in the context of small and medium enterprises

(SMEs) (Gresse, Punter, and Anacleto 2003), since the very factors that are prominent in these companies may actually become the cause of the problems. Some of these characteristics are:

- *Resource, schedule, and cost limitations:* The investment of 30 to 200 hours of resource effort in generating a measurement program is apparently worth more than the benefit to be obtained. The relationship between an investment in measurement and the business return is attenuated and complex (Berry, Jeffery, and Aurum 2004). This is one of the main reasons for a refusal to implement measurement programs, particularly in small settings where the scope of projects and maintenance services is smaller and easier to manage. In addition, these companies may not be able to contract software measurement experts or provide software measurement training.
- *Poor software measurement knowledge:* There is usually no measurement culture in these settings. Effective software measurement requires the documentation of a lot of information, models, and decisions. Thus, it is difficult for people who do not have extensive experience in software measurement (Briand, Differding, and Rombach 1996). In addition, the organization's members are unaware of the benefits of a measurement program; they think of measurement as a means of controlling staff, and they do not trust the benefits of these initiatives.

Since encouraging and implementing measurement programs in these kind of companies is difficult, and since these companies have characteristics that are particular to them, it is worth adapting measurement program methodologies to their specific reality. These companies require lightweight methodologies, whose roles could be adapted to company size and that include support modules for their basic measurement needs. This would make definition easier and make up for the deficiency in measurement knowledge (Laporte, Alexandre, and Renault 2008). Some of the proposed support modules are:

- To provide guidelines for the support of process improvement needs. These guidelines could suggest the measurement goals that support process improvement goals.
- To provide guidelines with which to develop the measurement goals and to understand the benefits and potential for management.
- To provide guidelines for the integration of measurement into the software processes.
- To provide guidelines as to how to adapt the measurement definition to the company's measurement maturity.
- To provide measurement examples.

This article introduces MIS-PyME, a framework for defining measurement programs based on software indicators. It is tailored to SMEs, in order to support the definition of successful measurement programs based on their characteristics. The article focuses on one of the support modules of the MIS-PyME framework, namely the MIS-PyME software measurement goals table, whose aim is to help SMEs define measurement programs that match their process improvement needs and maturity. This article also reports on how this module has been used to carry out a measurement program in a medium-sized company with a low capability level regarding its measurement maturity.

MIS-PyME FRAMEWORK

MIS-PyME is a methodological framework that focuses on the definition of measurement programs based on software indicators in small and medium settings. This section presents a brief description of MIS-PyME. A complete description of the framework can be found in (Diaz-Ley, Garcia, and Piattini 2008a).

MIS-PyME is focused on software development and maintenance companies or units with the typical characteristics of SMEs regarding measurement activities, namely:

- The people involved in the measurement program, including the measurement analyst, are from within the company and are not necessarily experts.
- The company has poor measurement maturity, that is, poor measurement culture, knowledge, and training. The measures collected in the company are few and the measurement process is not established in the unit or company, or it does not exist.
- The personnel are reluctant to use measurement.

- A small or medium software development company or unit with limited resources and fewer than approximately 50 people.

The MIS-PyME framework is classified in three main modules: 1) the methodology and roles (MIS-PyME methodology); 2) the work products that support the methodology (MIS-PyME measurement goals table, which is the subject of this article; MIS-PyME indicator template; and MIS-PyME database); and 3) the measurement maturity model (MIS-PyME measurement maturity model). The result is a measurement program definition that is composed of a number of measurement elements—indicators, measures, value criteria, and so on—and a measurement process that defines the activities required to carry out the measurement program. Figure 1 illustrates the MIS-PyME methodology framework.

The MIS-PyME methodology is based on the goal question metric (GQM) approach (Solingen and Berghout 1999) and the goal question indicator metric (GQ(I)M) approach (Park, Goethert, and Florac 1996; Goethert and Siviy 2004), but it is designed to define basic indicators that are commonly used and required in most small and medium software development settings. The steps of the MIS-PyME methodology are shown in Figure 2.

MIS-PyME uses work products that make definition and implementation easier and more reliable. These are:

- *The MIS-PyME measurement goals table*: The MIS-PyME framework proposes a set of structured measurement goals that usually are required in order to implement improvement activities related to software processes. This work product constitutes the main theme of this article and is described in depth in a later section.
- *MIS-PyME indicator templates*: An indicator template is defined for each measurement goal. The indicator template is designed to guide the measurement analyst in developing the measurement goal, to get from goals to measures, and to define the analysis required to achieve the goal. It also guides the measurement analyst in his or her implementation of the measurement program by integrating the indicators (and related measures) into the organization's processes. An indicator template shows, among other things, the possibility of implementing the indicator with regard to the measurement maturity of the company. It also outlines both the conditions for successfully implementing the indicator with regard to the previously required indicators and the measures required to build the indicator, and how to integrate this indicator into the software process. The typical questions that the indicator tries to answer are proposed. Typical outcomes and their related analysis also may be described and show the user what the indicator's potential is, and so on.
- *MIS-PyME database*: Each MIS-PyME indicator template contains examples of real indicators that have been defined in a successfully implemented measurement program.

FIGURE 1 MIS-PyME framework

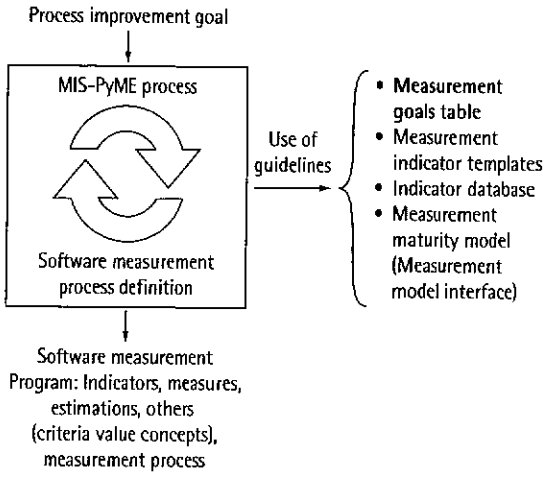
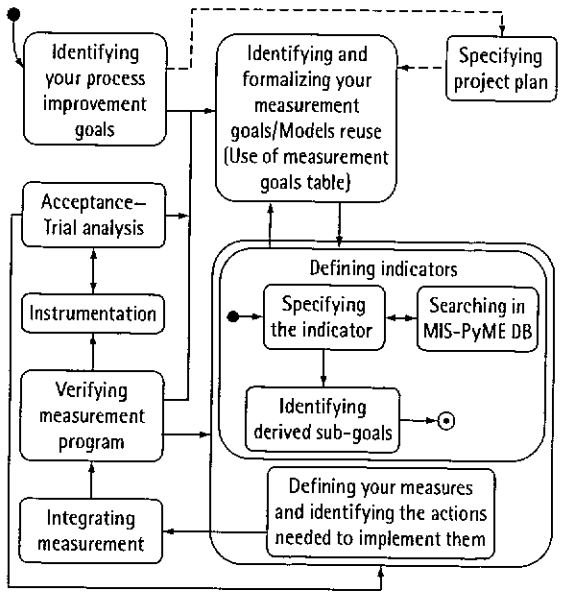


FIGURE 2 MIS-PyME methodology



The roles included in MIS-PyME are: the measurement analyst, who is responsible for defining and implementing the measurement program (it is preferable if his or her usual work relates to management, configuration management, quality, or security issues, rather than to design or development tasks); the top manager, who is the person with the most at stake in the measurement program and is also the promoter of the initiative (he or she might be the person responsible for processes management); and the reviewer, who is responsible for the verification and acceptance tasks (this role is usually played by project managers or experienced software developers). MIS-PyME methodology is designed so that only the top manager and the measurement analyst define the measurement program. Once a draft of the measurement program exists, it is verified and accepted by the reviewers, with the measurement analyst's assistance. Such a design is based on the assumption that managers' needs do not differ greatly from those of others. The top manager understands the company's needs and its way of working, and the aim is to seek common useful measurement goals that support software process improvement, although it is not possible to assign many resources to measurement initiatives. In addition, experience (Diaz-Ley, Garcia, and Piattini 2008a) has confirmed that this is a good practice in this context, since people are initially reluctant and reviewers make more suitable suggestions and are more motivated if a draft of the measurement program has already been defined.

The MIS-PyME measurement maturity model (MIS-PyME MMM) was designed to guide the user in defining the measurement program that is best suited to the company's maturity. This measurement model is based on the model proposed by Daskalantonakis, Yacobellis, and Basili (1990) and the Measurement Capability Maturity Model (M-CMM) defined by Niessink and Van Vliet (1998). MIS-PyME MMM defines a set of four themes (software management, quality and development capability, measurement scope, and tools support and measurement support for management issues) aligned to five maturity levels. Figure 3 shows a scheme representing the meaning of each measurement level.

Each theme focuses on a measurement aspect upon which the success of the software measurement depends and contains the description of the conditions that are necessary to attain a certain maturity level. Figure 4 shows an excerpt of the measurement maturity model.

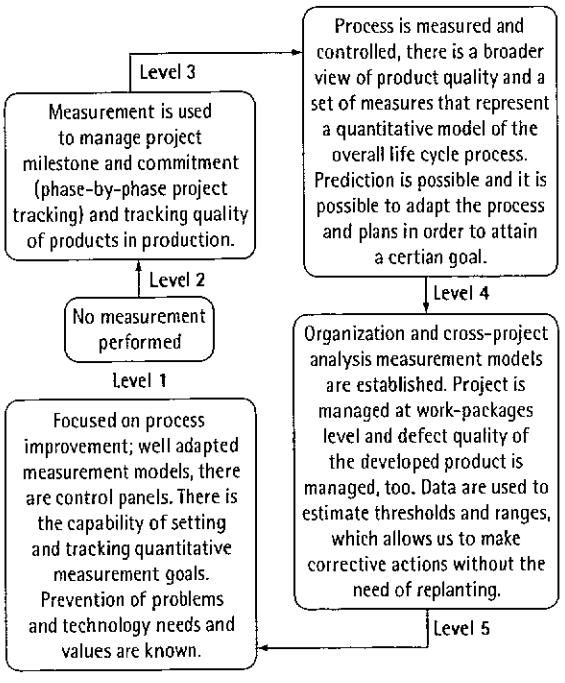
A questionnaire to assist the measurement analyst in discovering whether the company's measurement maturity is sufficient to implement certain indicators also is provided. A more detailed description of the MIS-PyME measurement maturity model can be found in (Diaz-Ley, Garcia, and Piattini 2008b). The benefits of using this measurement maturity model with regard to the popular CMMI (2006) are that the CMMI model includes a key process area called measurement and analysis. This process area defines good practices to implement a measurement process in an organization in order to reach capability level 2. CMMI (Team 2002) deals with most of the measurement aspects. However, they are distributed across most of the key process areas: software project planning at level 2, integrated software management at level 3, quantitative process management at level 4, and so on (Weber and Layman 2002), but this information is not dealt with in a separate module.

The following sections provide a more detailed description of one of the most relevant products of MIS-PyME: the measurement goals table.

MIS-PyME SOFTWARE MEASUREMENT GOALS TABLE

The goal of this MIS-PyME work product is to provide both the measurement analyst and the manager with

FIGURE 3 Capability level of MIS-PyME measurement maturity model



suggestions about the measurement goals that may help fulfill process improvement goals. This will help the user to comply with two of the main principles of MIS-PyME. The first is that of defining measurement programs that support the software process improvement initiatives, as is the case of the work of (Daskalantonakis 1992). This principle helps to define a measurement program with a firm organizational goal, as is the case of process improvement. Measurement is thus understood to be not an end but a means. The second principle deals with defining and implementing measurement programs that are adapted to the measurement maturity of the setting. Companies may work on defining and implementing measurement programs that they are able to successfully implement. They may not, however, attempt to achieve the best measurement when there are various impediments or obstacles that make a successful implementation impossible. These obstacles may be related to development, management or quality capability, support tools, and so on.

Identifying the measurement goals that the measurement program should fulfill is one of the first and most important activities in the definition of a measurement program. When the measurement goals are not well defined and do not represent what is required, the definition effort may be wasted, or even worse, the results of the measurement program analyses may be counterproductive. The analyses specified for the measurement program are based on the measurement goals defined. The measure specifications and the data that must be collected are, moreover, focused on performing the required analysis in order to satisfy the measurement goal defined.

The MIS-PyME software measurement goals table shows a set of common process improvement goals and proposes a set of measurement goals related to each process improvement goal that may help to achieve this. Figure 5 shows an excerpt of the MIS-PyME measurement goals table in which the measurement goals are classified in two ways. The first is based on the entities to which the measurement goal is applied. These entities are:

- PRJ: The entity to be measured is the project. Therefore, the attributes of the project, such as the duration, cost, effort, and so on, are measured.
- PROC: The entity to be measured is the process applied in the project. Therefore, the attributes related to the process applied in the project in order to develop software, such as conformance, effectiveness, efficiency, and so on, are measured.
- PROD: The entity to be measured is the software product. Therefore, attributes such as its quality are measured.
- PRJORG: The indicator analyzes information related to all the projects in the organization, making cross-project analyses.
- PROCORG: The indicator analyzes information regarding the processes applied in the organization's projects.

FIGURE 4 MIS-PyME measurement maturity model

Theme	Level 1	Level 2	Level 3	Level 4	Level 5
Software management, quality, and development capability	Immature processes. Projects depend on experienced professionals.	Repeat tasks that have been mastered in the past. Project depends on experienced professionals.	Projects characterized and reasonably understood. Project and development system management focus.	Measuring over process and process control. Focus on controlling the process.	Optimized process. Focus on process improvement.
Measurement scope	Carried out occasionally with experienced people or not at all.	Measurement is based on phase-by-phase project tracking.	Organization establishes standard processes and measurement models that are followed in projects and products.	Measurement is used in most of the projects and products.	Well-adapted measurement models. Measuring overall process.
Tools support	There are no tools to explicitly support process measurement.	Measurement support tools focusing on projects.	Project and product focus measurement tools. There is a measurement database for storing historical data.	Measurement support tools focusing on projects and products.	There are organizational tools that automatically . . .
Measurement support for management issues	Management is not supported by measures.	Basic project management. Milestones and commitment management.	The product developed is controlled by means of measures that are used to make decisions.	It is possible to predict the product, service, and other attributes before the . . .	It is possible to predict and prevent problems.

- PRODORG: The indicator analyzes information with regard to the entirety of the organization's products in order to derive general problems and improvement areas.

The other classification is based on the maturity level required to successfully implement the measurement goal. The measurement maturity level required for each software measurement goal is assigned based on the MIS-PyME MMM (see Figure 4). These levels are assigned different colors in order to highlight them and to ensure that users take note of what is being done. Level 2 measurement goals are colored in blue, level 3 measurement goals in green, level 4 in pink and gray (the gray indicators are usually implemented after the pink ones), while level 5 measurement goals are not colored.

FIGURE 5 An excerpt from the MIS-PyME software measurement goals table

Cat.	Measurement goal	Derived measurement goals
Improving project planning		
Improving project tracing and oversight		
PRJ	[blue] Monitoring the conformity of the project's main activities (or phase-by-phase) against what is planned, in order to understand the progress of the project and take reactive action if necessary.	[blue] Monitoring the requirements that have passed the development process's main activities (analysis, design, construction, acceptance, and so on).
		[blue] Monitoring the conformity of the duration of the main activities against the duration of the planned activity.
		[blue] Monitoring the conformity of the real effort against the planned effort per main activity.
		[blue] Monitoring the number of accepted open incidents during the software development activities in order to understand the remaining work.
		[blue] Monitoring the number of accepted change requests opened during the software development activities in order to understand the remaining work.
	[green] Monitoring the conformity of the project work-packages against what is planned (checking the thresholds) and helping to take corrective action if deviations exist.	Monitoring the requirements that have passed the development process activities (analysis, design, construction, acceptance, and so on) related to the work packages. . . .
	[green] Monitoring the "earned value" of the project in order to understand (checked by thresholds) the progress of the project in terms of schedule and cost and take corrective action.	[green] Characterizing the budgeted cost of work scheduled.
		[green] Characterizing the budgeted cost of work performed.
		[green] Characterizing the actual cost of work performed.
	[blue] Monitoring the cost progress in terms of budgeted and expended costs in order to understand the cost progress and taking reactive actions.	[blue] Characterizing the expended cost.
This process improvement goal (improve project tracking and oversight) is related to the improved product quality goal, specifically those measurement goals related to PRJ.		
Improving the process		
PROC	[green] Characterizing the process compliance in the project.	[green] Characterizing whether the project has performed each of the software processes activities defined in the development process of the company and if negative, understanding its consequences.
	[pink] Characterizing the process effectiveness in order to understand it.	[pink] Characterizing the defects detected in each activity of the process related to reviewing or testing the software products.
		[pink] Characterizing the rework performed in each activity of the software development process
	[pink] Characterizing the process efficiency in order to understand it.	[pink] Characterizing the productivity in each activity of the software development process.
	[gray] Evaluating the process effectiveness.	[gray] Evaluating the defects detected in each activity of the process related to review or testing the software products.
		[gray] Evaluating the rework performed in each activity of the software development process.
	[gray] Evaluating the process efficiency.	[gray] Evaluating the productivity in each main activity of the software development process.
	[pink] Characterizing the process effectiveness for each work-product in order to understand it.	[pink] Characterizing the rework performed in each activity of the software development process for each work product.
		[pink] Characterizing the defects detected in each activity of the process related to reviewing or testing the software products for each work product.
	[pink] Characterizing the process efficiency for each work product . . .	[pink] Characterizing the productivity at each activity of the software development process for each work product . . .

The measurement goals are defined by the following fields based on (Solingen and Berghout 1999) and (Park, Goethert, and Florac 1996):

- Purpose: Why the object will be analyzed, that is, characterization, monitoring, evaluation, prediction, control, and change.
- Focus of the indicator: This states the particular attribute of the object under study that will be characterized, evaluated, predicted, monitored, controlled, or changed. Examples of quality focuses are cost, reliability, correctness, defect removal, changes, user friendliness, maintainability, and so on.
- Entity to be measured: Target of the measurement activity, such as process (PROC), project (PRJ), and product (PROD).
- Usefulness: Final purpose of the indicator.
- Point of view: Description of the audience for whom the indicator is intended.
- Context: The environment in which the measurement will be carried out, analyzed, and interpreted. This also determines how the results can be generalized.

The fields that determine the suitable measurement maturity level for the measurement goal are the purpose, entity, and focus fields. Figure 6 shows the relationship between these fields and the themes of the measurement maturity model.

MIS-PyME MEASUREMENT GOALS TABLE: A CASE STUDY

This section shows how the MIS-PyME measurement goals table was applied in an experience that consisted of implementing a measurement program in the software development and maintenance department of Sistemas Técnicos de Loterías del Estado (STL), which has 38 employees. This company was created by the

FIGURE 5 An excerpt from the MIS-PyME software measurement goals table (Figure 5 continued)

Cat.	Measurement goal	Derived measurement goals
Improving product quality—Reliability		
PROD	[blue] Characterizing the reliability delivered to the client.	[blue] Characterizing the number of faults in the developed product, once delivered to the client.
		[blue] Characterizing the defects of the product delivered and in production.
	[green] Monitoring the reliability of the product developed during the project (checking the thresholds).	[green] Monitoring the density of failures during the test phases in order to make decisions about whether the next development phase can go ahead (checking the thresholds).
	[pink] Evaluating the reliability of the product delivered.	[pink] Evaluating the number of faults in the developed product once delivered to the client.
		Evaluating the defects of the product delivered and in production.
	Characterizing the fault tolerance.	
Other process improvement goals related to product quality are improving product quality - improving maintainability, usability of the product, product efficiency. These goals are not shown, for reasons of space.		
Improving the maintenance service		
PRODOR	[pink] Monitoring the mean time to repair and take reactive action (taking into account the product domain).	[pink] Monitoring the mean time for preventive maintenance.
		[pink] Monitoring the mean time for adaptative maintenance.
		[pink] Monitoring the mean time for corrective maintenance.
		[pink] Monitoring the mean time for corrective urgent maintenance.
	[blue] Evaluating the mean time to repair and improve the maintenance actions . . .	[blue]Evaluating the mean time for preventive maintenance . . .
Other process improvement goals are improving maintenance planning and client satisfaction.		
Level 5 does not need a support table, since at this maturity level the company knows exactly what to measure and improve in order to control their processes and improvement initiatives. The goals pursued at this stage are related to (Weber and Layman 2002): • Optimizing process performance based on goals. As an example of this: Improving defect detection efficiency of the testing process by 15 percent. • Optimizing business results from many small improvements. As an example, gradually increasing productivity by 8 percent as a result of 22 process improvements. • Optimizing business results from "individual" planned improvement efforts. Improving the product mean time between failures by 70 percent as a result of a planned process improvement effort that addressed the life cycle.		

Spanish Government and provides operational and information technology (IT) development services for the national lottery.

An experience and a case study were performed in order to define a required measurement program in a small division of a medium-sized company and to validate the MIS-PyME methodology. The experience and the case study are described in full in (Diaz-Ley, Garcia, and Piattini 2008a). The experience provided an idea of the usefulness and benefits of the proposed MIS-PyME measurement goals table in SMEs.

For this experience, one of the people in the department whose usual work was not related to software measurement was appointed as measurement analyst. The "top manager" was the director of this department and promoted the initiative. Among the four goals stated, one of the process improvement goals of the measurement program was to improve project and process monitoring and control.

The reviewers were people in the company who played the roles of project managers. No mention is made, however, of the reviewers in this section, since the MIS-PyME goals table is used in the first activities of the methodology and the reviewers do not take part in these activities.

Following MIS-PyME, the steps in which the measurement goals table work product was used were:

- *Step 1: Identifying your process improvement goals.* One of the process improvement goals stated by the top manager was "improving project and process monitoring and control." This process improvement goal matches that which is indicated in the MIS-PyME measurement goals table as "improve project tracing and oversight" (see Figure 5).
- *Step 2: Formalizing measurement goals and checking whether a measurement model is reused.* Measurement goals are specified in this step. The object of study, the purpose, the focus, and the environment are then defined. The MIS-PyME measurement goals table must help the user identify the measurement goals that will support the process improvement goals. In this step one also checks whether a measurement goal with the same description already exists in the organization's measurement process. If that is the case, its corresponding indicator template is checked to understand whether it fulfills the needs that support the

FIGURE 6 Indicator template fields that depend on measurement maturity

Indicator Field	MIS-PyME-MM theme	Description
Purpose	Software management, quality, and development capability	Measurement process has to fit with the rest of the processes. Otherwise, the implementation of the measurement program will in all probability fail. For example, one cannot measure the effectiveness between test phases if the test phases are not well differentiated.
	Measurement scope	There are certain kinds of measures that require a certain degree of measurement maturity and previous experience. For example, one cannot make reliable predictions on a particular aspect when there has not been any previous, frequent, and rigorous measurement of that aspect.
	Tools support	In order to implement some measurement programs, some tools are required, such as databases, tools that make it possible to visualize an indicator control panel, and so on.
	Measurement support for management issues	Measurement should be established to support process improvement goals, which also means management goals. If there is not any purpose in analyzing measurement in terms of decision making or corrective actions, the implementation of a measurement program is not recommended (for example, it is not advisable to implement a measurement program for project monitoring purposes). If the existing measurement data are not used to take simple corrective actions, it is not recommended to do so for other purposes such as optimization.
Entity	Measurement scope	If organizational information based on measurement is needed it is usually required to measure projects or products individually beforehand. Projects are the first entities to be measured; products come second and processes third.
Focus	Tool support	There are a number of measurements that cannot be performed if certain management or development tools are not in place.
	Processes capability	The aspect to be measured has to be established by the other development, management, or quality processes.

process improvement goal. STL, however, did not have any measurement program for this process improvement goal. The measurement analyst checked "improve project tracing and oversight" of the MIS-PyME measurement goals table. As the measurement maturity of the company was quite low, she decided to start with the measurement goals that were easy to implement, which were those in level 2 colored in blue, as shown in Figure 5. She then asked the top manager which measurement goals he considered to be relevant and they selected all of them, with the exception of the last one. The "improving product quality - reliability" area also was taken into account, as was suggested by the MIS-PyME measurement goals table and, consequently, the "monitoring the reliability of the product developed during the project" measurement goal was specifically included.

Figure 7 shows the final indicators defined for this process improvement goal. The same steps were followed for the other process improvement goals.

FIGURE 7 Measurement program indicators defined for the "improving project and process monitoring and control" process improvement goal

Indicator Name	Description
Ind-prj-prodcod: Monitoring the project progress at the coding phase	This shows the progress of the coding phase by showing the number of requirements coded in contrast with the total (ind-prj-reqcod), the progress in schedule (ind-prj-durationconformance), and in effort (Ind-prj-effortconformace).
Ind-prj-progverif: Monitoring the project progress at the verification phase	This shows the number of requirements verified with regard to the total number of requirements (ind-prj-reqverif), the progress of the verification phase by showing the number of open incidences and their severity (ind-prj-incpv), the progress in schedule (ind-prj-durationconformance), and in effort (Ind-prj-effortconformace).
Ind-progacept: Monitoring the project progress at the acceptance phase	This shows the number of requirements accepted with regard to the total number of requirements (ind-prj-reqacept), the number of open incidences and their severity (ind-prj-incpv), the progress in schedule (ind-prj-durationconformance), the defect density (ind-proj-denfailures), and in effort (Ind-prj-effortconformace).

Lessons Learned

In contrast with other previous experiences, the measurement program was defined more efficiently, effectively, and accurately when using the MIS-PyME measurement goals table, and it was successfully implemented. The nonexpert measurement analyst appreciated the measurement framework provided by the MIS-PyME, which guided her in asking the top manager questions to derive the measurement goals that would support the process improvement goal. She estimated that it was three times more efficient than GQ(I)M, the methodology used previously. This is because when using GQ(I)M the measurement goals may frequently be modified, since there is no guide upon which to base the questions, to identify the issues related to the process improvement, to identify the purpose of the goals, and so on. Some of the problems identified in the previous experience using GQ(I)M that MIS-PyME helps to avoid are explained next.

The measurement goals table prevented the analyst from defining measurement goals that would not be possible to successfully implement if the company's measurement maturity was not sufficient. This makes the final measurement program more reliable and eliminates a cause of failure in previous measurement experiences. For example, in a previous measurement program, there was one indicator related to the evaluation of the effectiveness of the development process that measured the reliability of the product under development in each test activity and compared it to a fixed goal. This indicator was rejected once it was implemented. The MIS-PyME measurement goals table indicated that the effectiveness of the process measurement goal corresponded with level 4 of measurement maturity. Consequently, it was not possible to determine the reliability that a product under development should have in each test activity based on the characteristics of the product developed, nor was it possible to understand the effects that the development process had on the product. This is one example of what a company that is inexpert in measurement might try to achieve when defining its first measurement programs, and how the MIS-PyME measurement goals table helped prevent this from happening.

The MIS-PyME measurement goals table also permits the measurement analyst and top manager to grasp the whole context. It does this by showing them the most common measurement goals related to the process improvement goal that may prevent analysts from defining other measurement goals that might not have any

precise relationship with the process improvement goal that they are dealing with. This problem also was dealt with in previous experiments in which the MIS-PyME framework was found to be quite effective.

RELATED WORKS

The idea of matching software measurement programs to software process improvement initiatives was supported by the GQM approach (Solingen and Berghout 1999) and in (Daskalantonakis 1992). However, none of the methodologies completely provides a support module for this issue.

Among the most representative methodology frameworks for software measurement, the following deserve special attention: the GQM approach (Rombach 1991; Basili 1992; Solingen and Berghout 1999; Basili and Weiss 1984); GQ(I)M, a goal-driven software measurement guidebook (Park, Goethert, and Florac 1996) that provides an extension of GQM to explicitly support the definition of indicators (Goethert and Siviy 2004); and ISO/IEC 15939 (ISO/IEC 2002), which identifies the activities and tasks necessary to successfully identify, define, select, apply, and improve software measurement under a generic project or the measurement organization structure.

Another related work is the GQM lightweight method defined by Gresse, Punter, and Anacleto (2003). This approach gives guidelines for adapting GQM to SMEs. It also mentions integrating the reuse of context-specific quality and resource models into the GQM model.

CMMI (2006) provides a specific process area to guide the measurement process, which is known as the measurement and analysis process area. Its purpose is to develop and sustain a measurement capability that is used to satisfy management information needs.

These proposals focus solely on the methodological aspect, that is, on the steps required to define and implement a measurement program. They do not deal with the supporting modules, nor do these frameworks provide specific guidelines concerning how to achieve the defined activities. Therefore, they do not provide specific information to assist the user in defining the measurement goals that support common process improvement goals.

The most complete framework is the Practical Software and Systems Measurement (PSM) framework (DoD 2000), which provides support for the definition of measurement programs. Parts 3 and 5 of this guide contain indicators and measures that are frequently used in projects. These indicators and measures are classified in issues and categories so as to guide the user in selecting the right measures with regard to his or her needs. There also is a template for each of the measures. This carefully describes the measure, its use, how to integrate it, the related attributes, and so on. Its goal is to provide project and technical managers with the best practices and guidelines in software measurement. Although PSM (DoD 2000) is undoubtedly a useful means to select the right measures for supporting common project issues, the MIS-PyME measurement goals table work product provides some benefits over and against PSM, such as:

1) The MIS-PyME software measurement goals table relates software measurement goals to software process improvement goals. As mentioned previously, MIS-PyME and other methodologies such as GQM (Solingen and Berghout 1999) consider that measurement programs should support process improvement, and this is not an end in itself but rather a means to achieve a clear goal for the process. The MIS-PyME software measurement goals table places the user in this context and guides him or her toward the definition of a measurement program with the process improvement goal in mind. PSM, however, only relates measures to project issues, so the scope of MIS-PyME is wider, since it specifies the indicators required in an organization that develops and maintains software products. There are some indicators that measure individual projects, (PRJ), others that make cross-projects analyze in order to understand the general tendency in the organization (PRJORG), and yet others that are related to all of the organization's products in a quest to understand the value and other characteristics of the services provided in general (PRODORG).
2) PSM does not address information regarding the maturity required to implement certain indicators. It addresses information regarding the constraints on the type of analysis (for example, part 5) or on the tools required (for example, part 2), and so on, but it does not relate these constraints to a measurement maturity model in which the limitations and evolution are clearly described. Moreover, it

lacks information regarding the constraints of the purpose of some indicators and their scope.

CONCLUSIONS AND FURTHER RESEARCH

This article addresses support for the definition of software measurement programs in SMEs by focusing on one of the support modules of the MIS-PyME methodology: the measurement goals table. This work product provides basic measurement goals that may help fulfill process improvement goals, which is the starting point of a measurement program. It consists of a table in which the most common process improvement goals are related to the software measurement goals that support the attainment of the defined process improvement goals. Furthermore, the definition of measurement goals is in line with the measurement maturity required by the company to successfully implement these goals. This is a key support module for SMEs, since these companies are not usually experienced in software measurement and do not have enough resources or a sufficient budget to be able to afford to contract measurement experts, provide training, or assign many resources to measurement issues.

The idea of bringing the measurement program into step with process improvement goals has been addressed by other methodologies and authors such as (Solingen and Berghout 1999) or (Daskalantonakis 1992), but to the best of the authors' knowledge, only MIS-PyME provides a measurement goals table as a support module for its methodology. The benefits of having this support module integrated into the methodology are that it makes the measurement program definition easier, more efficient, and more reliable. Measurement analysts save time in defining measurement goals, since they have a "process improvement goals measurement goals table" as a basis. It is more efficient, since the likelihood of being wrong in defining the measurement goals that will support process improvement goals is lower. It is also more reliable, since the measurement goals table helps the analyst to define measurement goals that have been adapted to the maturity of the company.

The MIS-PyME measurement goals table was applied in a case study of a small unit and the successful results were:

- The measurement goals definition of the measurement program was around three times more efficient than in previous experiences using GQ(I)M.
- It was more accurate, since the measurement goals were in line with the measurement maturity of the company.
- It was more effective, since it prevented analysts from defining other measurement goals that might not have had any precise relationship with the process improvement goal that they were dealing with.

In previous experiences, other measurement goals that were not suited to the maturity of the company were defined but then unsuccessfully implemented. It was therefore concluded that the MIS-PyME measurement goal table was quite useful.

MIS-PyME provides a number of guidelines that may influence managers and measurement analysts when defining predefined indicators. The MIS-PyME framework, however, is based on the assumption that managers' needs do not differ greatly from those of others. In SMEs, moreover, the cost of defining a measurement program from scratch is laborious and company members are not usually well trained enough to perform these tasks.

The authors' future work will involve testing and improving both the MIS-PyME measurement goals table and the other MIS-PyME support modules. A further goal of the authors' future research is to validate MIS-PyME's suitability for different software application domains.

Acknowledgment

The authors would like to thank the staff of Sistemas Técnicos de Loterías del Estado (STL) for their collaboration. This research has been sponsored by the COMPETISOFT (CYTED, 506AC0287), ESFINGE (Dirección General de Investigación del Ministerio de Educación y Ciencia, TIN2006-15175-C05-05), and INGENIO (Junta de Comunidades de Castilla-La Mancha, PAC08-0154-9262) projects.

REFERENCES

Basili, V. R. 1992. Software modeling and measurement: The goal/question/metric paradigm. Technical Report, CS-TR-2956. College Park: University of Maryland.

Basili, V. R., and D. Weiss. 1984. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering* 10, no. 11:758-773.

Berry, M., R. Jeffery, and A. Aurum 2004. Assessment of software measurement: An information quality study. In Proceedings of the 10th International Symposium on Software Metrics (METRICS'04). New York: IEEE.

Briand, L. C., C. M. Differding, and H. D. Rombach. 1996. Practical guidelines for measurement-based process improvement. *Software Process—Improvement and Practice* 2, no. 4: 253-280.

CMMI. 2006. CMMI for development, v1.2: Improving processes for better products. Pittsburgh: Software Engineering Institute.

CMMI Product Team. 2002. CMMI for systems engineering/software engineering, version 1.1: Staged Representation, CMU/SEI-2002-TR-002, ADA339224. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Daskalantonakis, M. K. 1992. A practical view of software measurement and implementation experiences within Motorola. *IEEE Transactions on Software Engineering* 18, no. 11:998-1010.

Daskalantonakis, M. K., R. H. Yacobellis, and V. R. Basili. 1990. A method for assessing software measurement technology. *Quality Engineering* 3, no. 1:27-40.

Diaz-Ley, M., F. García, and M. Piattini. 2008a. Implementing a software measurement program in SMEs: A suitable framework. IET Proceedings Software. 2(5), no. 5:417-436.

Diaz-Ley, M., F. García, and M. Piattini. 2008b. MIS-PyME software measurement maturity model: Supporting the definition of software measurement programs. In *Product Focused Software Development and Process Improvement* (PROFES'08), Rome, Italy. Germany: Springer/Heidelberg.

DoD. 2000. PSM: Practical software and systems measurement: A foundation for objective project management, version 4.0c. Washington, DC: U.S. Department of Defense (DoD).

ESSI. 1996. The CEMP project. ESSI project number 10358. Customized establishment of measurement programs, final report.

Goethert, W., and J. Siviy. 2004. Applications of the indicator template for measurement and analysis. Software Engineering Measurement and Analysis Initiative, SEI.

Gresse, C., T. Punter, and A. Anacleto. 2003. Software measurement for small and medium enterprises. In *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering (EASE)*, Keele, UK.

ISO/IEC. 2002. ISO/IEC 15939. Software Engineering—Software Measurement Process. Geneva, Switzerland: International Organization for Standardization.

Laporte, C. Y., S. Alexandre, and A. Renault. 2008. Developing international standards for very small enterprises. *IEEE Computer* 41, no. 3:98-101.

Niessink, F., and H. V. Vliet. 1998. Towards mature measurement programs. *Software Maintenance and Reengineering.*

Park, R. E., W. B. Goethert, and W. A. Florac. 1996. *Goal-driven software measurement-A guidebook.* Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Rombach, H. D. 1991. Practical benefits of goal-orientated measurement. In *Software Reliability and Metrics.* The Netherlands: Elsevier Applied Science.

Solingen, R. v., and E. Berghout. 1999. *The goal/question/metric method: A practical guide for quality improvement of software development.* London: McGraw-Hill.

Weber, C., and B. Layman. 2002. Measurement maturity and the CMM: How measurement practices evolve as processes mature. *Software Quality Professional* 4, no. 3.

BIOGRAPHIES

Maria Díaz-Ley has a master's degree in computer science and a master's degree in software measurement methodologies from the Universidad Autónoma de Madrid (UAM). She currently works for Sistemas Técnicos del Estado (STL). Her main responsibilities involve coordinating the IT area for international projects and organizing and developing integration and system tests. Furthermore, she is in charge of the STL software development process. As a researcher she belongs to the Alarcos Research Group (Universidad de Castilla La Mancha) and is pursuing a doctorate in software measurement methodologies for SMEs in the frame of the COMPETISOFT research program. She can be reached at maria.diaz@stl.es.

Félix García received his master's and doctorate degrees in computer science from the University of Castilla-La Mancha (UCLM). He is currently an associate professor in the Department of Information Technologies and Systems at the UCLM. He is member of the Alarcos Research Group and his research interests include business process management, software processes, software measurement, and agile methods.

Mario Piattini has a master's and a doctorate degree in computer science from the Technical University of Madrid. He is currently a professor in the Department of Computer Science at the University of Castilla-La Mancha (UCLM) in Ciudad Real, Spain. Piattini is the author of several books and papers on software engineering, databases, and information systems, and he leads the Alarcos Research Group of the Department of Information Systems and Technologies at UCLM. His research interests include software process improvement, database quality, software metrics, software maintenance, and security in information systems. He can be reached at mario.piattini@uclm.es.